# Models/Payment.js

import mongoose from "mongoose";

const paymentSchema = new mongoose.Schema({

  order: { type: mongoose.Schema.Types.ObjectId, ref: "Order", required: true },

  amount: { type: Number, required: true },

  paymentMethod: {

    type: String,

    enum: ["cash", "card", "online"],

    required: true,

  },

  status: {

    type: String,

    enum: ["pending", "completed", "failed"],

    default: "pending",

  },

  createdAt: { type: Date, default: Date.now() },

});

export default mongoose.model("Payment", paymentSchema);

# Services/paymentService.js

import Payment from "../models/Payment.js";

const confirmPayment = async (data) => {

  return await Payment.create(data);

};

export default { confirmPayment };

# utlis/Khalti.js

import axios from "axios";

async function requestKhalti({

  returnUrl,

  websiteUrl,

  totalAmount,

  orderId,

  orderName,

  customerInfo: { name, email, phone },

}) {

  const requestBody = {

    return\_url: returnUrl,

    website\_url: websiteUrl,

    amount: totalAmount,

    purchase\_order\_id: orderId,

    purchase\_order\_name: orderName,

    customer\_info: {

      name,

      email,

      phone,

    },

  };

  try {

    const response = await axios.post(

      process.env.KHALTI\_API\_URL,

      requestBody,

      {

        headers: {

          Authorization: `key ${process.env.KHALTI\_LIVE\_SECRET\_KEY}`,

        },

      }

    );

    return response.data;

  } catch (error) {

    return error.response?.data;

  }

}

export default requestKhalti;

# api/.env

APP\_URL=http://localhost:3000

CLOUDINARY\_API\_KEY=725995271339734

CLOUDINARY\_API\_SECRET=P83k3DgYN3132tIqQcoou-PNaLE

CLOUDINARY\_CLOUD\_NAME=dmenev3ev

KHALTI\_LIVE\_SECRET\_KEY=011b24f9ed75474aa30fb1eb569b1401

KHALTI\_API\_URL=https://dev.khalti.com/api/v2/epayment/initiate/

JWT\_SECRET=12345678910

MONGO\_URI=mongodb://127.0.0.1:27017/fpos

PORT=5000

DEADLINE="2025-11-11"

# Pages/Orders/Payment.jsx

import { confirmOrder } from "../../api/order";

import { ORDERS\_ROUTE } from "../../constants/routes";

import { useEffect, useState } from "react";

import { useNavigate, useParams, useSearchParams } from "react-router-dom";

import Spinner from "../../components/Spinner";

const OrderPayment = () => {

  const [loading, setLoading] = useState(true);

  const [error, setError] = useState(null);

  const [searchParams] = useSearchParams();

  const navigate = useNavigate();

  const params = useParams();

  async function getOrderConfirmation() {

    try {

      const orderId = params.id;

      const status = searchParams.get("status");

      const transactionId = searchParams.get("transaction\_id");

      await confirmOrder(orderId, status, transactionId);

    } catch (error) {

      setError(error.response.data);

    } finally {

      setLoading(false);

      setTimeout(() => {

        navigate(ORDERS\_ROUTE);

      }, 1500);

    }

  }

  useEffect(() => {

    getOrderConfirmation();

  }, []);

  return (

    <div className="py-8 px-2 sm:p-10">

      {loading ? (

        <div className="rounded-2xl bg-slate-100 p-10">

          <h3 className="text-center text-xl md:text-3xl text-textColor">

            Verifying payment

          </h3>

          <div className="flex items-center justify-center p-10">

            <Spinner height="h-10" width="w-10" />

          </div>

        </div>

      ) : error ? (

        <div className="rounded-2xl bg-red-100 p-10">

          <h3 className="text-center text-xl md:text-3xl text-textColor">

            Payment failed

          </h3>

          <div className="flex items-center justify-center p-10">{error} </div>

        </div>

      ) : (

        <div className="rounded-2xl bg-green-100 p-10">

          <h3 className="text-center text-xl md:text-3xl text-textColor">

            Payment success

          </h3>

        </div>

      )}

    </div>

  );

};

export default OrderPayment;

# app/.env

VITE\_API\_URL=http://localhost:5000

VITE\_PORT=3000

VITE\_APP\_URL=http://localhost:3000